

Lab Manual: 08

Lab Topic: Inheritance and Polymorphism

Course Code: CSE1116 (Object Oriented Programming Laboratory)

Course Instructor: Mir Moynuddin Ahmed Shibly, Lecturer, CSE, UIU.

Lab Activities

A. Defining the Superclass

(The *Account* class)

Design a class named **Account** that contains:

- A private **int** data field named **id** for the account (default **0**).
- A private **double** data field named **balance** for the account (default **0.0**).
- A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0.0**).
- A private **Calendar** data field named **dateCreated** that stores the date when the account was created. Use **Calendar** type object.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id, initial balance and annual interest rate. Within the constructor, assign the value of **dateCreated** using **Calendar.getInstance()**.
- The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
- The accessor method for **dateCreated**.
- A method named **getMonthlyInterestRate()** that returns the monthly interest rate. **monthlyInterestRate** is **annualInterestRate / 12**. Note that **annualInterestRate** is a percentage, e.g., like 4.5%.
- The method **getMonthlyInterestAmount()** is to return the monthly interest amount, not the interest rate. Monthly interest amount is **balance * monthlyInterestRate**.
- A method named **withdraw** that withdraws a specified amount from the account.
- A method named **deposit** that deposits a specified amount to the account.

Write a test program that creates an Account object with an account ID of 1122, a balance of \$20,000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw \$2,500, use the deposit method to deposit \$3,000, and print the balance, the monthly interest, and the date when this account was created.

B. Creating the Subclasses

Create two subclasses for checking and saving accounts named as **CheckingAccount** and **SavingsAccount**.

- A checking account has an overdraft limit which is double type variable.
- A savings account has issued with a credit card automatically. It holds the 16-digit card number an expiry date (Calendar type object). SavingsAccount class must have a method `getCreditBalance()` that returns a credit balance which is three times of the current balance in the account.

C. Defining an Array/ArrayList of Account type objects in Main method

Polymorphism allows us to refer subtype objects using a super type variable.

To understand polymorphism, you need to define an array/array list of Account type objects based on user-provided option. Your main() method must display the following first:

```
Press (1) for creating a Checking Account
Press (2) for creating a Savings Account
```

You must create at least 4 account type objects in this manner and perform one deposit and one withdraw operation for each account.

Afterwards, print the followings for each account using the concept of Polymorphism.

| | |
|---|--|
| <p>For a Checking Account:</p> <p>This is a Checking Account</p> <p>Account ID:</p> <p>Date Created:</p> <p>Current Balance:</p> <p>Annual Interest Rate:</p> <p>Monthly Interest Amount:</p> <p>Overdraft Limit:</p> | <p>For a Savings Account:</p> <p>This is a Savings Account</p> <p>Account ID:</p> <p>Date Created:</p> <p>Current Balance:</p> <p>Annual Interest Rate:</p> <p>Monthly Interest Amount:</p> <p>Credit Card Number:</p> <p>Card Expiry Date:</p> <p>Credit Balance:</p> |
|---|--|

D. (Additional Task) Defining abstract class and abstract methods

Abstract class cannot have any objects.

Declare the **Account** class as an abstract class and all methods of Account as abstract methods except constructors. Modify the rest of the code accordingly.